



Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

CRT-2003-18

**A CONSTRAINT-BASED APPROACH
TO THE GOLOMB RULER PROBLEM**

by

Philippe Galinier

Brigitte Jaumard

Rodrigo Morales

Gilles Pesant

August 2003

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone: (514) 343-7575 / Télécopieur: (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca





**PUBLICATION
CRT-2003-18**

**A CONSTRAINT-BASED APPROACH
TO THE GOLOMB RULER PROBLEM**

by

Philippe Galinier^{1,2}

Brigitte Jaumard^{1,3}

Rodrigo Morales^{1,2}

Gilles Pesant^{1,2}

¹ Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 (philipg, brigitt, rorro, pesant@crt.umontreal.ca)

² Département de génie informatique, École Polytechnique de Montréal, C.P. 6079, succursale Centre-ville, Montréal, Canada H3C 3A7

³ Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2003

Abstract

A Golomb ruler with m marks can be defined as a set of m distinct integers such that the differences between them are all distinct. Golomb rulers and related problems play an important role in, e.g., radio communication, VLSI architectures, convolutional self-orthogonal codes and radio-astronomy. An optimal Golomb ruler minimizes the value of the largest mark. Finding optimal Golomb rulers is a highly combinatorial problem and difficult to solve efficiently even for small sizes. Most successful algorithms essentially rely on backtracking. We propose a combination of constraint programming and improved lower bounds in order to solve it. In particular, we introduce a new lower bound based on minimum weight assignments and multi-decompositions. Experimental results show a significant improvement in the amount of backtracking required.

Keywords: Golomb ruler, constraint programming, minimum weight assignment, multi-decomposition.

Résumé

Une règle de Golomb avec m marques peut être définie comme un ensemble de m entiers distincts tel que les différences entre les deux marques sont toutes distinctes. Les règles de Golomb et leurs variantes jouent un rôle important par exemple en radio télécommunications, en architectures de VLSI, pour les codes correcteurs doublement orthogonaux et en radioastronomie. Une règle de Golomb est optimale si, pour un nombre fixé de marques, sa longueur est minimale. Trouver les règles de Golomb optimales est un problème fortement combinatoire et difficile à résoudre efficacement même pour des règles avec un petit nombre de marques. Les algorithmes les plus efficaces sont du type recherche en profondeur d'abord. Nous proposons une nouvelle approche basée sur une combinaison de programmation par contraintes et de bornes inférieures améliorées pour le résoudre. En particulier, nous introduisons une nouvelle borne inférieure basée sur l'affectation de poids de valeur minimale et sur une technique de multidécomposition. Des résultats expérimentaux montrent que des améliorations sensibles sont obtenues en termes de nombre de nœuds explorés dans la recherche en profondeur d'abord.

Mot-clés : règle de Golomb, programmation par contraintes, affectation de poids minimum, multidécomposition.

INTRODUCTION

A GOLOMB ruler with m marks can be defined as a set of m distinct integers $x_1 < \dots < x_m$ such that the $m(m-1)/2$ differences $d_{ij} = x_j - x_i$ are all distinct. The value x_m is referred to as the length of the ruler. An optimal Golomb ruler is such that, given m , its length x_m is minimum. Finding optimal Golomb rulers corresponds to a highly combinatorial problem, for which no exact algorithm can provide solution with a reasonable computing time for $m \geq 24$. However, using finite geometry tools, very good approximate values for optimal Golomb rulers can be computed for up to $m = 150$ [23]. Recent progress in the implementation and distributed computation of the exact algorithms confirmed that those values were indeed exact for $19 \leq m \leq 23$, see [1].

Most exact algorithms are backtracking algorithms, i.e., branch-and-bound algorithms with a depth first search strategy. They all make use of upper bounds which are set equal to the minimum length in the experiments. Lower bounds are usually simple ones, as sophisticated ones remains not so accurate and too costly to compute in a branch-and-bound scheme. One of the first efficient algorithms is due to Shearer [24] and was able to compute minimum Golomb rulers up to 16 marks. Later, Dollas, Rankin and McCracken [18] proposed the GVANT algorithm and managed to compute the optimal Golomb ruler with 19 marks. Using bitmaps to store mark locations and spans, as well as logical operations to perform their updates in a branch-and-bound scheme led to the design of the GARSP

algorithm. Then, with the help of "Distributed.Net" who had taken a liking to Golomb rulers, minimum Golomb rulers with $20 \leq m \leq 23$ were computed, see [1]. More recently, the OGR project is coordinating the search for optimal Golomb rulers with 24 and 25 marks.

Several lower bounds have been proposed in order to speed up the backtracking algorithms. Other techniques, semi-affine and projective planes originally proposed by Singer [26] have also been used to obtain good upper bounds. They have been improved by [15], [8], [7], [12], and used to define greedy and randomized heuristics [6] and to improve backtracking algorithms [25], [13]. Tabu search has been applied to this problem as well [10].

Constraint programming has also been used since it is often more efficient than classical approaches for very tightly constrained problems. Smith *et al.* [27] reformulated the Golomb ruler problem as a constraint programming problem. They studied different representations, several orderings for fixing marks as well as the introduction of redundant constraints. Although the resulting algorithm is not competitive with the GARSP algorithm for the computation times (due to the optimized data structures used in the latter), the authors obtain interesting results in terms of the number of nodes expanded in the branching scheme.

There are several other related problems such as Golomb Rectangles [22] and graceful graphs [11]; see Galian [9] for additional variants. Golomb rulers and related problems play an important role in, e.g., radio communication [3], VLSI architectures [16], convolutional self-orthogonal codes [21] [4], or radio-astronomy [5].

In this paper, we propose to explore further the constraint programming approach, trying in particular to introduce different and more efficient lower bounds. The paper is organized as follows. Section I presents several relevant bounds for this problem, some of which are used in our constraint programming algorithm described in section II. Experimental results are then presented in section III.

I. BOUNDS ON DISTANCES BETWEEN MARKS

In this section we present trivial and not so trivial lower and upper bounds on distances between marks that are key to the success of the constraint programming algorithm presented in section II.

A. Lower Bounds

An elementary lower bound for d_{ij} stems from the observation that d_{ij} can be decomposed as a sum of $j-i$ terms: $d_{ij} = d_{i,i+1} + \dots + d_{j-1,j}$. As these terms are positive integers and must all be different, we deduce that their sum is

no smaller than the sum of the first $j - i$ positive integers and so

$$d_{ij} \geq \Sigma_{j-i} \quad 1 \leq i < j \leq m \quad (1)$$

where $\Sigma_p = p(p+1)/2$. For example, for $(i, j) = (1, 5)$:

$$\begin{aligned} d_{15} &= d_{12} + d_{23} + d_{34} + d_{45}, \\ d_{15} &\geq \Sigma_4 = 10. \end{aligned}$$

We observe that the part of a ruler comprised between marks x_i and x_j must also exhibit the Golomb property. Therefore it must be of length at least ℓ_{j-i+1} where ℓ_q represents the minimal length of an q -mark Golomb Ruler. Knowledge of optimal Golomb rulers for smaller numbers of marks can thus be exploited:

$$d_{ij} \geq \ell_{j-i+1} \quad 1 \leq i < j \leq m. \quad (2)$$

Bound (1) above exploits the fact that the differences $d_{p,p+1}$ comprised between consecutive marks x_p and x_{p+1} are all different. But we note that not only consecutive differences but, more generally, all differences d_{pq} inside d_{ij} (called segments of d_{ij}) are different. We introduce now new kinds of decompositions that involve segments d_{pq} with not necessarily consecutive integers p and q ; moreover, we are going to consider sets of decompositions, called multi-decompositions, as introduced in [12] and developed in [13] to find best possible lower bounds. We present below three decompositions for $(i, j) = (1, 5)$:

$$\begin{aligned} d_{15} &= d_{12} + d_{23} + d_{34} + d_{45}, \\ d_{15} &= d_{13} + d_{35}, \\ d_{15} &= d_{12} + d_{24} + d_{45}. \end{aligned}$$

The number of occurrences of each segment d_{pq} (called its multiplicity) is given by the function μ : for the example above, $\mu(d_{12}) = 2, \mu(d_{23}) = 1, \mu(d_{34}) = 1, \mu(d_{45}) = 2, \mu(d_{13}) = 1, \mu(d_{35}) = 1, \mu(d_{24}) = 1$, and is zero everywhere else. There are two segments of multiplicity 2, and five segments of multiplicity 1. Therefore, as all segments are positive integers and are all different, we deduce:

$$3 \cdot d_{15} \geq 2 \cdot (1+2) + (3+4+5+6+7) = \Sigma_2 + \Sigma_7 = 3+28 = 31.$$

Hence $d_{15} \geq \lceil \frac{31}{3} \rceil = 11$. We observe that this new bound is stronger (greater) than that obtained before from $\Sigma_{5-1} = 10$.

More formally, we call *decomposition* of d_{ij} a set $(d_{a_t, b_t})_{t=1..T}$ such that $a_1 = i, b_T = j, a_t < b_t (1 \leq t \leq T)$ and $a_{t+1} = b_t (1 \leq t \leq T-1)$. All possible differences $d_{pq} (i \leq p < q \leq j)$ will be called segments. For commodity, the different segments will be numbered from 1 to R . The set of segments is denoted by $\mathcal{Y} = \{y_1, \dots, y_R\} = \{d_{pq} : i \leq p < q \leq j\}$.

Among the huge set of possible decompositions, we consider a family of decompositions denoted by $\mathcal{M}(i, j, k)$, $k = 1, \dots, m$. Roughly speaking, the underlying idea that makes these multi-decompositions promising is to generate as many segments with as few repetitions as possible. The multi-decomposition $\mathcal{M}(i, j, k)$ is defined by:

$$\mathcal{M}(i, j, k) = \bigcup_{v=1}^k \bigcup_{w=1}^v L(i, j, v, w)$$

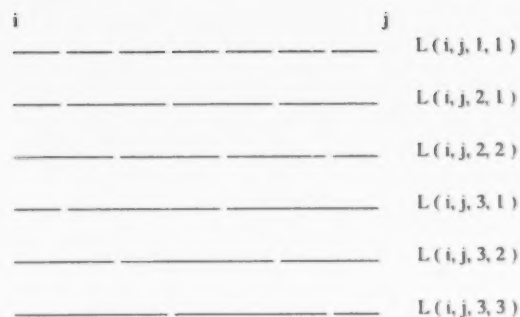


Fig. 1. Multi-decomposition $\mathcal{M}(i, j, 3)$

where

$$\begin{aligned} L(i, j, v, w) &= \{d_{i, i+w}\} \cup \bigcup_{\alpha=0}^{\alpha_{\max}-1} \{d_{i+w+\alpha \cdot v, i+w+(\alpha+1) \cdot v}\} \\ &\cup \{d_{i+w+\alpha_{\max} \cdot v, j}\} \end{aligned}$$

and

$$\alpha_{\max} = \max\{\alpha : i + w + \alpha \cdot v < j\}.$$

For example, figure 1 represents the multi-decomposition $\mathcal{M}(i, j, 3)$. Also note that the multi-decomposition of d_{15} presented before is $\mathcal{M}(1, 5, 2)$.

The number of decompositions in $\mathcal{M}(i, j, k)$ is easily seen to be $k(k+1)/2$. Let $\mu_{ijk} : \mathcal{Y} \rightarrow \mathbb{N}$ denote the multiplicity function for $\mathcal{M}(i, j, k)$. The bound deduced from multi-decomposition $\mathcal{M}(i, j, k)$ is

$$d_{ij} \geq \mathcal{M}_{ijk} = \lceil \frac{\mathcal{M}'_{ijk}}{k(k+1)/2} \rceil, \quad 1 \leq i, j, k \leq m, i < j \quad (3)$$

where $\mathcal{M}'_{ijk}$ represents the minimal sum of the segments weighted by their multiplicity, taking into account that all segments are of different length:

$$\mathcal{M}'_{ijk} = \min\left\{\sum_{r=1}^R \mu_{ijk}(y_r) \cdot n_r : n_1, \dots, n_R \in \mathbb{N}^* \text{ and distinct}\right\}.$$

The greatest $\mathcal{M}_{ijk}$, $1 \leq k \leq m$, will be denoted $\mathcal{M}_{ij*}$. The values of μ_{ijk} depend only on $j - i$, k and m — they can be precomputed and stored. The computation of $\mathcal{M}_{ijk}$ presented in the above example can be generalized as follows. Suppose E_h represents the number of segments with multiplicity h and C_h the number of segments with a multiplicity greater or equal to h : $C_h = \sum_{h'=h}^{k(k+1)/2} E_{h'}$. $\mathcal{M}'_{ijk}$ is then given by the following equation:

$$\mathcal{M}'_{ijk} = \sum_{h=1}^{k(k+1)/2} \Sigma_{C_h}.$$

For example: if there are six segments with multiplicity 1 and three segments with multiplicity 4, E_h and C_h have the following values:

h	1	2	3	4	5
E_h	6	0	0	3	0
C_h	9	3	3	3	0

and $\mathcal{M}'_{ijk} = \Sigma_9 + 3\Sigma_3$.

B. Upper bounds

We can derive an upper bound on d_{ij} from the fact that $d_{pq} = d_{p,p+1} + d_{p+1,p+2} + \dots + d_{i-1,i} + d_{ij} + d_{j,j+1} + \dots + d_{q-2,q-1} + d_{q-1,q}$ for $p < i < j < q$. In particular, this is true for d_{1m} and if we let $x_1 = 0$ (see (12) in section II-B) we may simplify and rewrite the expression as $d_{ij} = x_m - (d_{12} + \dots + d_{i-1,i} + d_{j,j+1} + \dots + d_{m-1,m})$. The parenthesized sum is made up of $m - 1 - j + i$ distances which, as remarked before, must total at least $\Sigma_{m-1-j+i}$. We therefore have:

$$d_{ij} \leq x_m - (m - 1 - j + i)(m - j + i)/2, \quad 1 \leq i < j \leq m. \quad (4)$$

Also, we have considered an upper bound using ℓ_q lower bounds. Then, our new upper bound on d_{ij} is:

$$d_{ij} \leq x_m - (m - 1 - j + i)(m - j + i)/2, \quad 1 < i < j < m, \quad (5a)$$

$$d_{1j} \leq x_m - \ell_{m-j+1}, \quad 1 < j < m, \quad (5b)$$

$$d_{im} \leq x_m - \ell_i, \quad 1 < i < m. \quad (5c)$$

C. Tightening the bounds under some assumptions

We now make two assumptions that enable us to derive tighter bounds. In section II, we will validate these assumptions and use the results to improve the efficiency of our algorithm.

Assumption (a): We are given a set $\mathcal{F} \subset \mathbb{N}^*$ of forbidden values to which the length of any segment cannot belong.

We reconsider the previous bounds in light of this first assumption. Bound (1) is weaker than (2), which is based on optimal Golomb rulers, but our assumption opens the door to another that is not dominated by it. Let $\Sigma_q^{\mathcal{F}}$ represent the sum of the q smallest values among $\mathbb{N}^* - \mathcal{F}$. Our lower bound becomes

$$d_{ij} \geq \Sigma_{j-i}^{\mathcal{F}} \quad 1 \leq i < j \leq m. \quad (6)$$

Similarly let $\ell_q^{\mathcal{F}}$ represent the minimal length of a q -mark ruler that does not produce any distances in $\mathcal{F}$. We get

$$d_{ij} \geq \ell_{j-i+1}^{\mathcal{F}} \quad 1 \leq i < j \leq m. \quad (7)$$

Note that such a bound is tighter but may be much harder to obtain than the previous one. The multi-decomposition lower bound is easily extended to

$$d_{ij} \geq \mathcal{M}_{ijk}^{\mathcal{F}} = \left\lceil \frac{\mathcal{M}_{ijk}^{\mathcal{F}}}{k(k+1)/2} \right\rceil \quad 1 \leq i, j, k \leq m, \quad i < j \quad (8)$$

with $\mathcal{M}_{ijk}^{\mathcal{F}} = \min\{\sum_{r=1}^R \mu_{ijk}(y_r) \cdot n_r : n_1, \dots, n_R \in \mathbb{N}^* - \mathcal{F} \text{ and distinct}\}$. The greatest $\mathcal{M}_{ijk}^{\mathcal{F}}$, $1 \leq k \leq m$, will be denoted $\mathcal{M}_{ij*}^{\mathcal{F}}$. The equation for $\mathcal{M}_{ijk}^{\mathcal{F}}$ can be generalized to $\mathcal{M}_{ijk}^{\mathcal{F}}$ as we have:

$$\mathcal{M}_{ijk}^{\mathcal{F}} = \sum_{h=1}^{k(k+1)/2} \Sigma_{C_h}^{\mathcal{F}}.$$

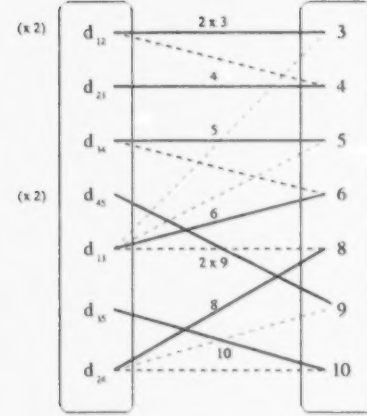


Fig. 2. Minimum assignment in G

The upper bound also has a natural extension:

$$d_{ij} \leq x_m - \Sigma_{m-1-j+i}^{\mathcal{F}} \quad 1 \leq i < j \leq m. \quad (9)$$

Assumption (b): For each segment y_r we are given $\text{Dom}(y_r)$, the set of possible lengths for y_r . Moreover, we suppose that $\text{Dom}(y_r) \subseteq \mathbb{N}^* - \mathcal{F}$. Therefore, (b) is stronger than (a).

This assumption makes sense for two of the previous bounds. Let ℓ_q^{Dom} represent the minimal length of a q -mark ruler whose segments exhibit a length belonging to their respective domain, as given by $\text{Dom}()$. We then have

$$d_{ij} \geq \ell_{j-i+1}^{\text{Dom}} \quad 1 \leq i < j \leq m, \quad (10)$$

$$d_{ij} \geq \mathcal{M}_{ijk}^{\text{Dom}} = \left\lceil \frac{\mathcal{M}_{ijk}^{\text{Dom}}}{k(k+1)/2} \right\rceil, \quad 1 \leq i, j, k \leq m, \quad i < j \quad (11)$$

with $\mathcal{M}_{ijk}^{\text{Dom}} = \min\{\sum_{r=1}^R \mu_{ijk}(y_r) \cdot n_r : n_r \in \text{Dom}(y_r), n_1, \dots, n_R \text{ distinct}\}$. The greatest $\mathcal{M}_{ijk}^{\text{Dom}}$, $1 \leq k \leq m$, will be denoted $\mathcal{M}_{ij*}^{\text{Dom}}$.

This time to compute $\mathcal{M}_{ijk}^{\text{Dom}}$, we introduce the weighted bipartite graph $G = (\mathcal{Y}, \bigcup_{r=1}^R \text{Dom}(y_r), E, w)$ where

- $E = \{(y_r, d) \in \mathcal{Y} \times \bigcup_{r=1}^R \text{Dom}(y_r) : d \in \text{Dom}(y_r)\}$
- $w : E \rightarrow \mathbb{N}, e = (y_r, d) \mapsto w(e) = \mu_{ijk}(y_r) \cdot d$

In this graph, an assignment represents a legal matching of lengths to the segments of $\mathcal{Y}$. The total weight associated with such an assignment is $\sum_{r=1}^R \mu_{ijk}(y_r) \cdot n_r$ for a certain choice of n_r 's. Therefore $\mathcal{M}_{ijk}^{\text{Dom}}$ equals the value of the minimum weight assignment in the graph. It is well known that a minimum weight assignment can be computed in polynomial time [2]. Figure 2 represents an assignment on graph G corresponding to the example presented above. Let the domains of the segments be $\text{Dom}(d_{1,2}) = \{3, 4\}$, $\text{Dom}(d_{2,3}) = \{4\}$, $\text{Dom}(d_{3,4}) = \{5, 6\}$, $\text{Dom}(d_{4,5}) = \{9\}$, $\text{Dom}(d_{1,3}) = \{3, 5, 6, 8\}$, $\text{Dom}(d_{3,5}) = \{10\}$, $\text{Dom}(d_{2,4}) = \{8, 9, 10\}$. The value of the minimum weight assignment is $\mathcal{M}_{1,5,2}^{\text{Dom}} = 2 \times 3 + 4 + 5 + 6 + 8 + 2 \times 9 + 10 = 57$, from which we get $\mathcal{M}_{1,5,2}^{\text{Dom}} = \left\lceil \frac{\mathcal{M}_{1,5,2}^{\text{Dom}}}{3} \right\rceil = \left\lceil \frac{57}{3} \right\rceil = 19$.

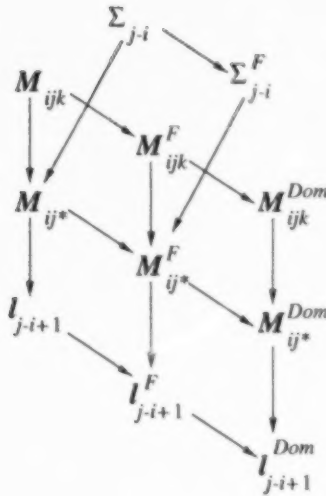


Fig. 3. Dominance relationships among the lower bounds

D. A taxonomy of lower bounds

It is clear that, under any assumption, the bound ℓ is stronger than M_* that is itself stronger than any M_k , whatever k . M_* is also stronger than $\Sigma = M_1$, which may or may not be weaker than M_k . Moreover, we recall that assumption (b) is stronger than (a). Figure 3 represents the different bounds in addition to the dominance relations between them.

II. A CONSTRAINT PROGRAMMING ALGORITHM

In this section, we propose a new algorithm, set within a constraint programming framework. We first present very briefly some main characteristics of constraint programming and refer the interested reader to other texts such as [17] for a more comprehensive introduction. Next, we present the model, including some instances of the bounds presented in section I. We also discuss operational issues such as level of consistency, variable- and value-selection heuristics.

A. Constraint Programming Approach

The application of constraint programming to the solution of combinatorial problems is based on modeling those problems using finite domain variables. For each such variable, the (finite) set of all possible values it may take, called its domain, is saved and updated as the computation progresses. Constraints are expressed about the combinations of values these variables may take and are enforced using specialized filtering algorithms, either built-in or user defined, that work by removing inconsistent values from the domains of variables. Ideally, the algorithm should remove all inconsistent values, that is, those for which no combination of values for the rest of the variables can be found that satisfies the constraint, but this cannot always be achieved with reasonable efficiency. The constraints (or, more precisely, their filtering algorithms) interact through shared variables (or, more precisely, their domains). Accordingly,

our model for the problem at hand is composed of such constraints.

This filtering out of inconsistent values will not be sufficient in general to obtain a solution directly. Search is still necessary, usually taking the form of a search tree whose branches correspond to fixing a variable to a value in its current domain. Many strategies for choosing the next variable to fix and the next value to try have been devised and are often problem dependent. At each node of the tree, more filtering can occur since at least one of the domains has shrunk (for the variable just fixed).

B. Model

The basis of our CP model was inspired by the work described in [27]. For an m -mark ruler, we define m finite domain variables x_i , $1 \leq i \leq m$, representing the position of each mark on the ruler. Given a maximum length L for the ruler, each of these variables has domain $\{0, 1, \dots, L\}$. In addition, $m(m-1)/2$ variables d_{ij} , $1 \leq i < j \leq m$, represent the difference between marks x_j and x_i . Their domain is implicitly defined as $\{1, 2, \dots, L\}$ given constraints (13) and (14) to follow.

We now describe the constraints. It is common to place the first mark, x_1 , at position 0:

$$x_1 = 0. \quad (12)$$

To avoid symmetries and thus reduce the solution space, marks should appear in lexicographic order on the ruler:

$$x_i < x_{i+1} \quad 1 \leq i \leq m-1. \quad (13)$$

Apart from avoiding symmetries, the latter ensures that differences d_{ij} will be positive as we link the two types of variables:

$$d_{ij} = x_j - x_i \quad 1 \leq i < j \leq m. \quad (14)$$

By definition of the problem, the differences between marks must be distinct. To the set of binary disequalities we could use to model this, we prefer the more powerful *alldifferent* constraint [20]. Its advantage over the former in this context has already been studied in [27].

$$\text{alldifferent}(d_{12}, d_{13}, \dots, d_{1m}, d_{23}, \dots, d_{m-1,m}). \quad (15)$$

Additional symmetry can be removed by not considering both a ruler and its mirror image:

$$d_{12} < d_{m-1,m}. \quad (16)$$

An alternate way to eliminate such symmetries is¹

$$d_{1,m/2} < d_{m+1-m/2,m}. \quad (17)$$

Another redundant constraint to consider is

$$d_{ij} = d_{ik} + d_{kj} \quad 1 \leq i < k < j \leq m. \quad (18)$$

¹as suggested independently by J.-F. Puget and [13]

Note that only constraints (13)-(15) are necessary to adequately model our problem² — the rest may be added to improve the efficiency of the algorithm, which is crucial in practice as we will soon find out.

Accordingly some of the bounds presented in section I have been included. First, we use the upper bound exactly as defined previously to produce binary constraints

$$d_{ij} \leq x_m - (m - 1 - j + i)(m - j + i)/2 \quad 1 \leq i < j \leq m.$$

Since optimal Golomb rulers for small m are known (currently up to 23), we can add unary constraints

$$d_{ij} \geq \ell_{j-i+1} \quad 1 \leq i < j \leq m, j - i + 1 \leq 23. \quad (19)$$

We do not include lower bound (1) since it is weaker, but rather use an instance of its improved version, (6), to be defined shortly.

C. Operational issues

Before going further with the model, let us describe the search algorithm. The algorithm fixes marks consecutively from x_1 to x_m and tries valid positions for them in ascending order; the computation of bounds and constraint propagation make it possible to reduce the size of the search tree. The algorithm is defined recursively by the function *explore* presented below:

explore(Dom, q, v, L)

1. $Dom(x_m) := Dom(x_m) \cap \{1..L - 1\}$
2. $Dom(x_q) := v$
3. if not propagate() return L
4. compute lower and upper bounds and reduce the domains accordingly
5. if not propagate() return L
6. if $q = m - 1$ return $\min(Dom(x_m))$
7. for each $w \in Dom(x_{q+1})$ do
 - (a) $L := \min(L, \text{explore}(Dom, q + 1, w, L))$
8. return L

Function *explore* describes the treatment applied at each node in the search tree. Its first parameter is the function $Dom : \{x_1..x_m\} \rightarrow 2^{\{0..L\}}$ that defines the current domains of the variables at the considered node. The two following parameters indicate the index q of the next variable to be instantiated and the value v to assign to x_q . Note that marks x_1 to x_{q-1} are already instantiated, hence each $Dom(x_j)$ ($1 \leq j \leq q - 1$) is a singleton that indicates the value assigned to x_j . The function returns the length of the shortest Golomb Ruler whose marks belong to the indicated domains and that is strictly inferior to parameter L , if there exists one; else, it returns L .

Function *propagate* performs constraint propagation. As a result, the current domains of some variables may be possibly reduced. At the end of this process, a domain value may be empty, indicating there exists no Golomb Ruler that extends the current instantiation; this case is called a failure. Function *propagate* returns *false* if a failure is

detected and *true* otherwise. For clarity, explicit calls to *propagate* were added in the algorithm's description. Note however that such calls are performed automatically in constraint programming, each time the domain of a variable is reduced.

First, mark x_q is fixed to v and constraint propagation is performed. Then, the upper and lower bounds are computed and the domains of the variables are reduced accordingly: if a lower bound LB (resp. an upper bound UB) is found for a particular x_j , then $Dom(x_j)$ is set to $Dom(x_j) \cap \{LB \dots \infty\}$ (resp. $Dom(x_j) \cap \{1 \dots UB\}$) and constraint propagation is performed. Note that value L is returned if a failure is detected. There are now two possible cases. The first one occurs if $q = m - 1$ and corresponds to a success: indeed, the current instantiation can be extended to a Golomb Ruler by assigning the value $\min(Dom(x_m))$ to x_m . If the second case occurs ($q < m - 1$), function *explore* is called recursively with all possible values $w \in Dom(x_{q+1})$ to be assigned to x_{q+1} . Note that L is updated in order to represent the length of the best solution found so far, each time a new better solution is found.

The search is initiated by *explore*($Dom_0, 1, 0, L$), where argument Dom_0 represents the initial domains and L an integer. Note that the function can easily be modified in order to exhibit the optimal ruler produced by the algorithm.

Our model includes two ternary constraints, (14) and (18), on which (generalized) arc consistency could be applied. This level of consistency verifies that for every value in the current domain of each variable involved in the constraint, there are some values in the domain of each of the other two variables that satisfy it. This notion of consistency is stronger than the bound consistency normally used with this type of constraint and which only reasons about the extremal values of domains.

D. More bounds

Some of the bounds come with a computational cost that may be prohibitive if applied to all pairs (i, j) . Consequently, we have so far restricted their use and synchronized it with the progression of the search strategy: we typically limit the application of a bound to the segment between the current mark and the last one. In the following, let x_q be the current mark to fix.

In order to profit from assumption (a), we define our forbidden set $\mathcal{F}$ as the distances between pairs of marks that have already been fixed, that is $\{d_{ij} : 1 \leq i < j \leq q - 1\}$. (6) thus becomes

$$d_{qm} \geq \sum_{m-q}^{\mathcal{F}}. \quad (20)$$

Using (7) is not as straightforward since the information required is not necessarily available. In fact, a large database of $\ell_m^{\mathcal{F}}$ exists for small values of m but we have not yet incorporated it [1].

It has been verified experimentally that for $j - i \leq 17$, the value of k that maximizes $\mathcal{M}_{ijk}$ is never above 3. Based

²and (15) can be modeled in a weaker form as discussed earlier

on this data, we decided to use

$$d_{qm} \geq \mathcal{M}_{qm}^{\mathcal{F}} \quad 1 \leq k \leq 3. \quad (21)$$

Meeting assumption (b) is no harder in our context since the constraint programming approach naturally maintains domains for the d_{ij} variables. As before we implemented a restricted version of (11):

$$d_{qm} \geq \mathcal{M}_{qm}^{Dom} \quad 1 \leq k \leq 3. \quad (22)$$

We will also experiment with a slightly more extensive version of (21) and (22) :

$$d_{im} \geq \mathcal{M}_{im}^{\mathcal{F}} \quad q \leq i \leq m-1, 1 \leq k \leq 3, \quad (23)$$

$$d_{im} \geq \mathcal{M}_{im}^{Dom} \quad q \leq i \leq m-1, 1 \leq k \leq 3. \quad (24)$$

III. EXPERIMENTAL RESULTS

The algorithm described in the previous section was implemented in ILOG Solver 4.4 [14] and tests were performed on a Sun Ultra-5 (400MHz, 128Mb of RAM). We experimented with several versions of the algorithm and give below the correspondance between algorithmic features and column headings in the various tables:

V_0 : model (12)-(16), (4), (19), (20)

V_1 : model (12)-(15), (17), (4), (19), (20)

V : model (12)-(15), (17), (5), (19), (20)

$\mathcal{F}$: lower bound (21)

D : lower bound (22)

$\mathcal{F}^*$: lower bound (23)

D^* : lower bound (24)

Ca : arc consistency on constraint (14)

Cb : arc consistency on constraint (18)

TABLE I

NUMBER OF BRANCHES EXPLORED TO FIND AN m -MARK RULER OF MINIMAL LENGTH L (F) OR PROVE THAT NO SHORTER ONE EXISTS (P).

m	L	S	V_0	V_1	V	$V + \mathcal{F}$	$V + \mathcal{F}^*$	$V + D$	$V + D^*$
8	34	F	19	19	19	19	18	19	18
		P	307	279	279	277	259	277	259
9	44	F	133	131	131	130	118	130	118
		P	1 325	1 029	1 029	971	925	970	924
10	68	F	665	503	503	479	395	479	395
		P	5 442	4 303	4 303	4 104	3 496	4 079	3 477
11	92	F	4 634	36 527	36 527	32 919	30 180	32 747	30 035
		P	187 050	105 280	105 280	93 831	85 018	93 177	84 517
12	86	F	226 941	82 015	82 015	71 037	60 641	70 393	60 158
		P	615 666	273 923	273 923	236 217	200 455	233 579	198 521
13	108	F	4 825 572	4 405 942	4 405 942	3 710 413	3 416 986	3 676 323	3 389 277
		P	17 327 217	5 658 378	5 658 378	4 720 341	4 346 128	4 670 309	4 306 887

TABLE II

CPU TIME IN SECONDS USED TO FIND AN m -MARK RULER OF MINIMAL LENGTH L (F) OR PROVE THAT NO SHORTER ONE EXISTS (P).

m	L	S	V_0	V_1	V	$V + \mathcal{F}$	$V + \mathcal{F}^*$	$V + D$	$V + D^*$
8	34	F	0.0	0.0	0.0	0.0	0.0	0.1	0.1
		P	0.2	0.2	0.2	0.2	0.2	0.6	0.7
9	44	F	0.1	0.1	0.1	0.1	0.1	0.3	0.4
		P	1.0	0.8	0.8	0.9	0.8	2.7	3.2
10	68	F	0.7	.5	0.5	0.5	0.4	1.7	1.9
		P	5.7	4.6	4.6	4.6	4.2	16.8	20.9
11	92	F	6.6	51.6	52.9	49.1	42.7	147.0	160.1
		P	300.7	159.7	154.0	151.0	132.4	461.0	524.7
12	86	F	406.0	154.1	156.0	142.0	117.5	493.0	521.0
		P	1 186.0	582.0	589.0	536.0	452.9	1 838.0	2 069.0
13	108	F	11 828.0	11 436.0	11 746.0	10 342.0	9 202.0	31 196.0	30 424.0
		P	46 148.0	15 055.0	15 462.0	13 491.0	12 073.0	41 012.0	49 207.0

The first two columns of tables I and II compare constraints eliminating symmetries. We see that constraint (17) is more efficient than constraint (16) both in the number of branches explored and the computation time. Third column shows version using upper bound (5), and we conclude that there are not differences using upper bound (5) or upper bound (4). Column $V + \mathcal{F}$ shows that bound (21) improves the results. Using the bound on unfixed marks (version $V + \mathcal{F}^*$ in table II) shows significant improvement both in the number of branches explored and the computation time at the same time: though not as important, it appears to increase with m . This is currently our best version. Our best bound, reported in the last two columns, improves further on the size of the search tree but our current implementation loses on the computation time.

TABLE III

NUMBER OF BRANCHES EXPLORED TO FIND AN m -MARK RULER OF MINIMAL LENGTH L (F) OR PROVE THAT NO SHORTER ONE EXISTS (P).

m	L	S	V	$V + Ca$	$V + Cb$	$V + D$	$V + D + Ca$	$V + D + Cb$	$V + D + Ca + Cb$	$V + \mathcal{F}$
8	34	F	19	18	15	19	18	15	15	15
		P	279	236	183	277	220	171	171	180
9	44	F	131	99	85	130	97	85	85	85
		P	1 029	789	719	970	730	677	677	691
10	68	F	503	413	297	479	331	247	247	266
		P	4 303	2 707	2 416	4 079	2 422	2 241	2 241	2 393
11	92	F	36 522	28 334	15 890	32 747	22 526	13 982	13 982	14 931
		P	105 280	80 709	50 553	93 177	62 734	45 391	45 391	47 812
12	86	F	82 015	55 549	41 582	70 393	44 160	34 503	34 503	39 063
		P	273 923	204 228	143 043	233 579	156 377	119 775	119 775	134 634

TABLE IV

CPU TIME IN SECONDS USED TO FIND AN m -MARK RULER OF MINIMAL LENGTH L (F) OR PROVE THAT NO SHORTER ONE EXISTS (P).

m	L	S	V	$V + Ca$	$V + Cb$	$V + D$	$V + D + Ca$	$V + D + Cb$	$V + D + Ca + Cb$	$V + \mathcal{F}$
8	34	F	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1
		P	0.2	0.3	0.6	0.6	0.8	1.0	1.2	0.6
9	44	F	0.1	0.2	0.4	0.3	0.4	0.6	0.7	0.4
		P	0.8	1.5	3.3	2.7	3.4	5.2	5.7	3.2
10	68	F	0.5	1.0	2.2	1.7	2.0	3.1	3.5	2.2
		P	4.8	8.4	22.0	16.8	18.7	32.5	35.9	21.6
11	92	F	52.9	97.8	219.0	147.0	168.9	264.4	301.0	212.9
		P	164.0	310.0	739.0	461.0	551.0	932.0	1 052.0	719.0
12	86	F	156.0	304.0	865.0	493.0	532.0	1 031.0	1 129.0	821.0
		P	589.0	1 175.0	3 502.0	1 838.0	2 233.0	4 266.0	4 707.0	3 333.0

Tables III and IV study the addition of arc consistency for the ternary constraints. It definitely reduces the search tree but its computational cost remains prohibitive. This confirms a finding of [27]. Note though that arc consistency on $d_{ij} = d_{ik} + d_{kj}$ dominates that on $d_{ij} = x_j - x_i$.

TABLE V

CPU TIME IN SECONDS AND NUMBER OF BRANCHES EXPLORED TO PROVE THE OPTIMALITY OF SOME KNOWN SHORTEST GOLOMB RULERS.

m	time	branches
12	128	57 788
13	2 813	1 141 762
14	30 343	9 017 621
15	404 622	112 956 246

In order to observe the behavior of the best version of the proposed algorithm on still larger instances, a parallel implementation using Parallel Solver [19], an extension of ILOG Solver, was run on four processors (Sun Ultra-2, 400MHz) in a shared memory environment. Table V reports CPU

time and number of branches explored to prove the optimality of the 12- to 15-mark Golomb rulers. Each additional mark appears to increase the time needed by roughly one order of magnitude. With respect to the sequential versions, the number of branches explored decreases significantly.

CONCLUSION

We proposed a combination of constraint programming and sophisticated lower bounds in order to solve the Golomb ruler problem. It improved on previous efforts at a constraint programming approach. As is often the case with CP, borrowing some known results and algorithms proved important. Most of the algorithmic ingredients were analyzed experimentally. We also presented a taxonomy of lower bounds for this problem and some new bounds were designed especially for a constraint-based approach.

However, one must not forget that other approaches have so far been more successful in tackling this problem, though they rely heavily on a database of ℓ_m^F for small rulers. On the one hand, our algorithm is five times better than the best algorithms from the literature, such as GARSP, in terms of the number of branches explored. On the other hand, it is 300 times slower, mostly because it does not use optimized data structures such as the bitmap in GARSP.

We nevertheless believe that still more could be done with this approach. Foremost, an optimized special-purpose implementation of constraint propagation for the Golomb ruler problem should greatly speed up the algorithm while still benefitting from the productive algorithmic ideas of this approach. Also, a careful study of which pairs (i, j) or triplets (i, j, k) are more productive for our ternary constraints and some of our bounds is currently being undertaken. Instead of posting constraints for all pairs or triplets in a brute force fashion, a thoughtful selection could maintain about the same level of pruning while significantly reducing the computation time. Other bounds mentioned in section I have yet to be exploited. Finally, the question of whether marks are best fixed in lexicographic order has not been definitely settled.

REFERENCES

- [1] "In Search of the Optimal 20, 21 & 22 Golomb Rulers," <http://members.aol.com/golomb20/wem23.htm>.
- [2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, 1993.
- [3] W.C. Babcock, "Intermodulation interference in radio systems," *Bell System Technical Journal*, pp. 63-73, 1953.
- [4] C. Cardinal, D. Haccoun, F. Gagnon, and N. Batani, "Turbo Decoding Using Convolutional Self Doubly Orthogonal Codes," ICC'99 - 1999 IEEE International Conference on Communications, pp. 113-117, 1999.
- [5] W.E. Carter, D.S. Roberson, J.E. Pettay, B.D. Tapley, B.E. Shultz, R.J. Eanes, and M. Lefeng, "Variations in the rotation of the earth," *Science*, vol. 224, pp. 957-961, 1984.
- [6] Y.M. Chee and C.J. Colbourn, "Constructions for difference triangle sets," *IEEE Transactions on Information Theory*, vol. 43, pp. 1346-1349, 1997.
- [7] Z. Chen, "Further results on difference triangle sets," *IEEE Transactions on Information Theory*, vol. 40, pp. 1268-1270, 1994.
- [8] Z. Chen, P. Fan, and F. Jin, "Disjoint difference triangle sets," *IEEE Transactions on Information Theory*, vol. 38, pp. 518-522, 1992.
- [9] J.A. Galian, "A Survey: Recent Results, Conjectures, and Open Problems in Labeling Graphs," *J. Graph Theory*, vol. 13, pp. 491-504, 1989.
- [10] P. Galinier and B. Jaumard, "A tabu search algorithm for difference sets and golomb rulers," submitted for publication, 2002.
- [11] S.W. Golomb, "How to Number a Graph," in *Graph Theory and Computing*, R.C. Read, Ed., pp. 23-37. Academic Press, 1972.
- [12] P. Hansen, B. Jaumard, and C. Meyer, "On Lower Bounds for Numbered Complete Graphs," *Discrete Applied Mathematics*, vol. 94, pp. 205-225, 1999.
- [13] S.C. Hoa, "Construction de règles de Golomb optimales," M.Sc. thesis, École Polytechnique de Montréal, 1999.
- [14] ILOG S.A., Gentilly, France. *ILOG Solver Reference Manual*, version 4.4., 1999.
- [15] T. Klove, "Bounds and construction for difference triangle sets," *IEEE Transactions on Information Theory*, vol. 35, pp. 879-886, 1989.
- [16] P. Lavoie, D. Haccoun, and Y. Savaria, "New VLSI Architectures for Fast Soft-Decision Threshold Decoders," *IEEE Transactions on Communications*, vol. 39, pp. 200-207, 1991.
- [17] K. Marriott and P.J. Stuckey. *Programming with Constraints: an Introduction*. MIT Press, 1998.
- [18] D. McCracken A. Dollas, W.T. Rankin, "A New Algorithm for Golomb Ruler - Derivation and Proof of the 19 Mark Ruler," *IEEE Transactions on Information Theory*, vol. 44, pp. 379-382, 1998.
- [19] L. Perron, "Search Procedures and Parallelism in Constraint Programming," *Proc. 5th Intl Conf. on Principles and Practice of Constraint Programming*, Springer LNCS vol. 1713, pp. 346-360, 1999.
- [20] J.-C. Régin, "A filtering algorithm for constraints of difference in CSPs," in *Proc. 12th Natl Conf. on Artificial Intelligence (AAAI-94)*, pp. 362-367, 1994.
- [21] J. P. Robinson and A. J. Bernstein, "A Class of Binary Recurrent Codes with Limited Error Propagation," *IEEE Transactions on Information Theory*, vol. 13, pp. 106-113, 1967.
- [22] J.P. Robinson, "Golomb rectangles," *IEEE Transactions on Information Theory*, vol. 31, pp. 781-787, 1985.
- [23] J.B. Shearer, "<http://www.research.ibm.com/people/s/shearer/dts.htm>."
- [24] J.B. Shearer, "Some new optimum golomb rectangles," *IEEE Transactions on Information Theory*, vol. 36, pp. 183-184, 1990.
- [25] J.B. Shearer, "Some New Difference Triangle Sets," *Journal of Combinatorial Mathematics and Combinatorial Computing*, vol. 27, pp. 65-72, 1998.
- [26] J. Singer, "A theorem in finite projective geometry and some applications to number theory," *Transactions American Mathematical Society*, vol. 43, pp. 377-385, 1938.
- [27] B. M. Smith, K. Stergiou, and T. Walsh, "Modelling the golomb ruler problem," Workshop on Non-binary Constraints, IJCAI, 1999.